Designing an X.500 User Interface: One Year In

Andrew Findlay Damanjit Mahl Stefan Nahajski

Brunel University

x500@brunel.ac.uk

ABSTRACT

The outline of a Directory user interface design was presented at the 1989 UKUUG conference in Cardiff. Several working interfaces have now been built and released for public comment. A design has been finalised as a result of experience and feedback gained from the working interfaces. The evolution of this design is charted, and the current X Window System and MS-Windows implementations are described. An overview of the latest design is given, and progress is reported.

1. Introduction.

The idea of X.500 is ... *to facilitate the interconnection of information processing systems to provide directory services. The set of all such systems, together with the directory information which they hold, can be viewed as an integrated whole, called the Directory. The information held by the Directory, collectively known as the Directory Information Base (DIB), is typically used to facilitate communication between, with, or about objects such as application entities, people, terminals, and distribution lists*[1].

The X.500 Directory can be thought of as an ‘electronic phone book’. Like a phone book, a geographic structure is imposed on the data; no phone book lists all people in the world in a single alphabetic sequence. It is necessary to have some idea of where a person might be before starting to search for them.

The X.500 Directory is similar; information is held in a tree structure, called the Directory Information Tree (DIT). Information is distributed across a large number of co-operating Directory System Agents (DSAs), each holding data concerned with a relatively small area. A consequence of this is that any searches covering a wide geographic area tend to be slow and expensive. User interfaces must therefore encourage the user to narrow the field of search as rapidly as possible.

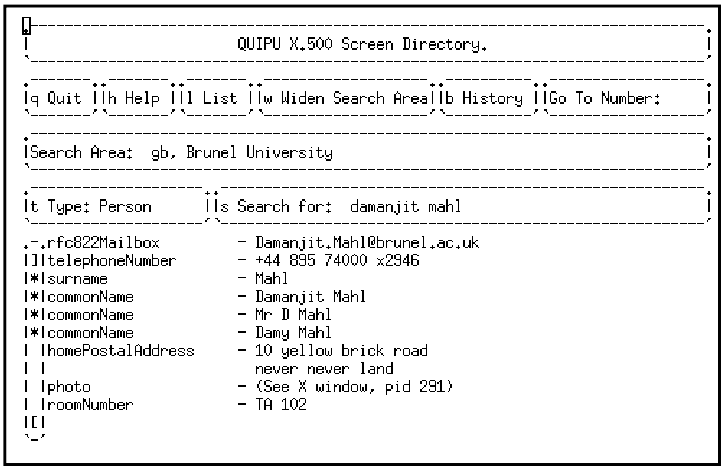
This paper describes the continuing design and implementation of user interfaces for the X.500 Directory.

2. Background

At the 1989 UKUUG conference in Cardiff, a paper was presented which introduced some of the services that an X.500 Directory would provide, and described an outline design of a user interface to make use of these services. Since then the team at Brunel University has produced four interfaces called sd, xd, xdsm and pod. Brief descriptions of these are given later. A design document has been written in the light of experience gained from those prototypes, and two new interfaces called Xdir and PCdir are currently under development.

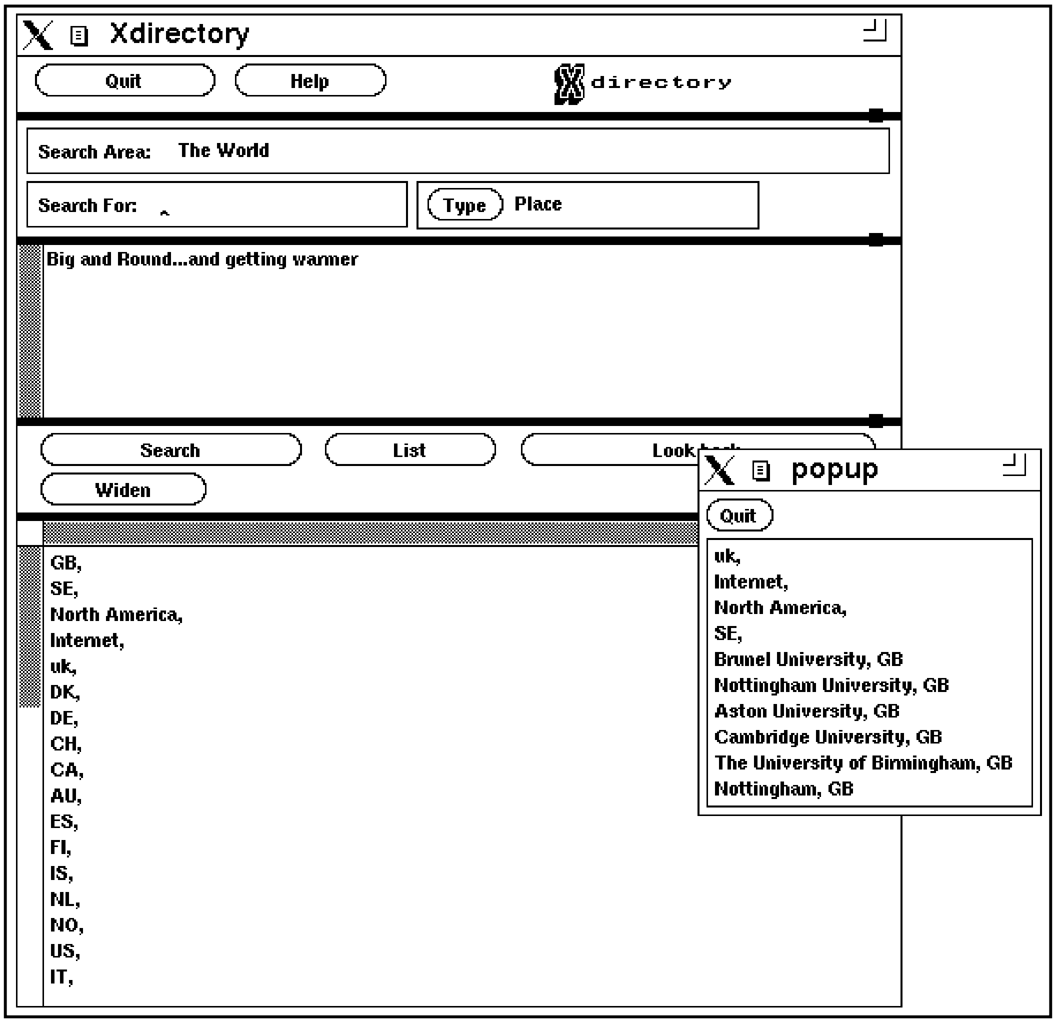
3. Current User Interfaces

Near the beginning of the year, an interface called sd was developed. This is a character mode interface for UNIX which derived from the early interfaces that were distributed as part of ISODE/QUIPU[2]. It provides a simple mechanism for navigating the Directory, and performing searches, listing children of a node etc. The user is presented with a screen similar to that shown. Pressing an appropriate key invokes an action.



It was decided at a meeting of the Directory Pilot Group that some prototype Directory user interfaces should be presented at the Networkshop 90 conference held at Newcastle University in March. This resulted in three interfaces being developed in addition to sd, namely xd, xdsm and pod. These were written for X and used the Xt toolkit and athena widget set (Xdsm initially used the Hewlett Packard widget set but switched to the athena widgets early on).

Xd and Xdsm occupied a large screen area, presenting information such as current Directory position, directory information about the current position and a list of children of the current position at the same time on a single form. In addition to this Xdsm maintained a list of 'places visited' which was shown at the bottom of the form. The interfaces were driven by pointing and clicking the cursor on a name in a list or on "buttons" which invoked directory or interface operations. Xd differed from Xdsm, by showing the history of places visited in



a separate pop-up, this being invoked by pressing a button. Both interfaces used pop-ups to implement help facilities. A screen shot of xd is shown below.

The idea of using pop-ups was taken further with the writing of Pod. Instead of maintaining an area in which a list of children of the current position could be displayed, Pod created a pop-up containing the list when required.

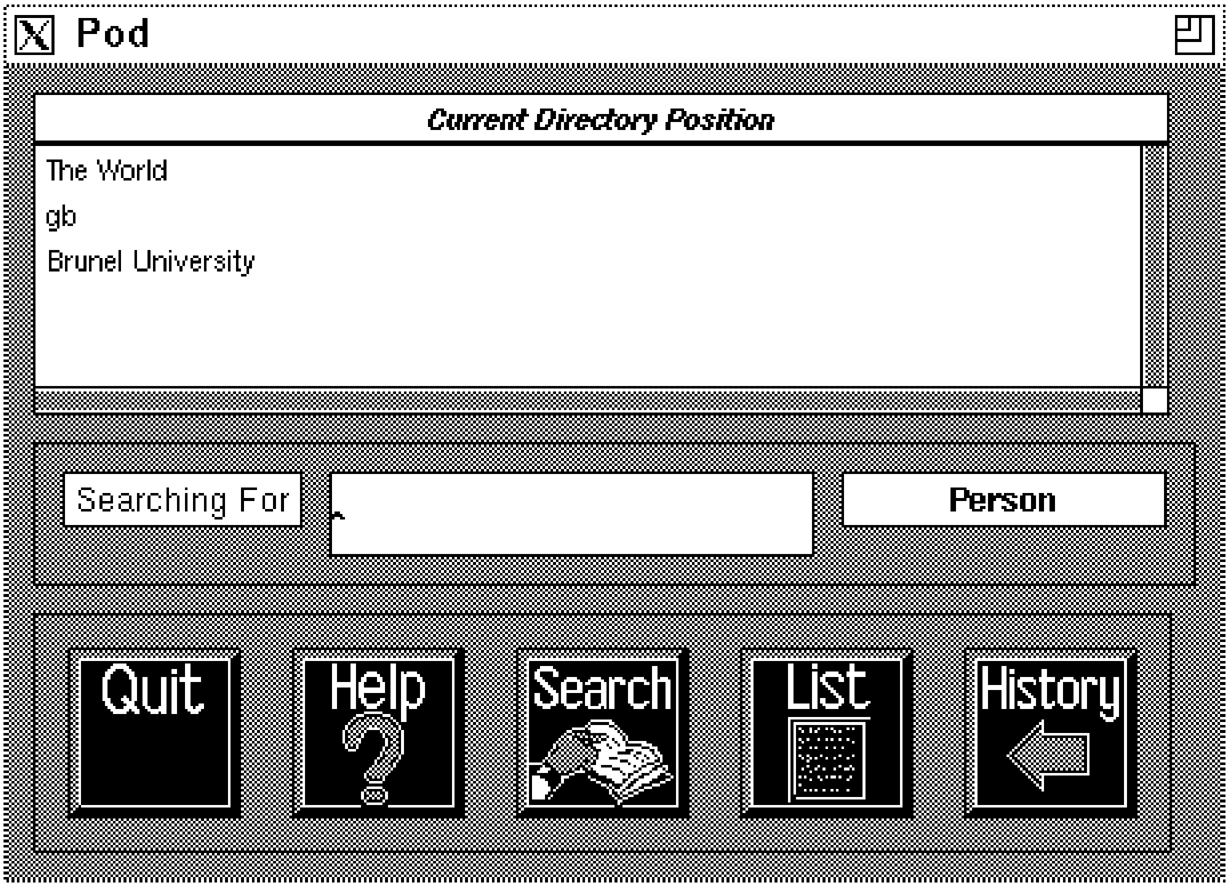
It was felt that Pod was a definite improvement over the previous interfaces and the decision was taken to further develop Pod.

4. Pod Today

Pod is now a high functionality DUA. It enables a user to read, modify, list and search the Directory. A search type defaulting mechanism is employed and complex search filters can be defined. Both the defaulting mechanism and search filters are configurable. All Directory results are shown in popup windows which can be maintained on screen, removed, or reused by later operations. A popup help card is also available, which has a 'cursor position sensitive' text. Examples of the features provided by Pod are shown in the following screen shots.

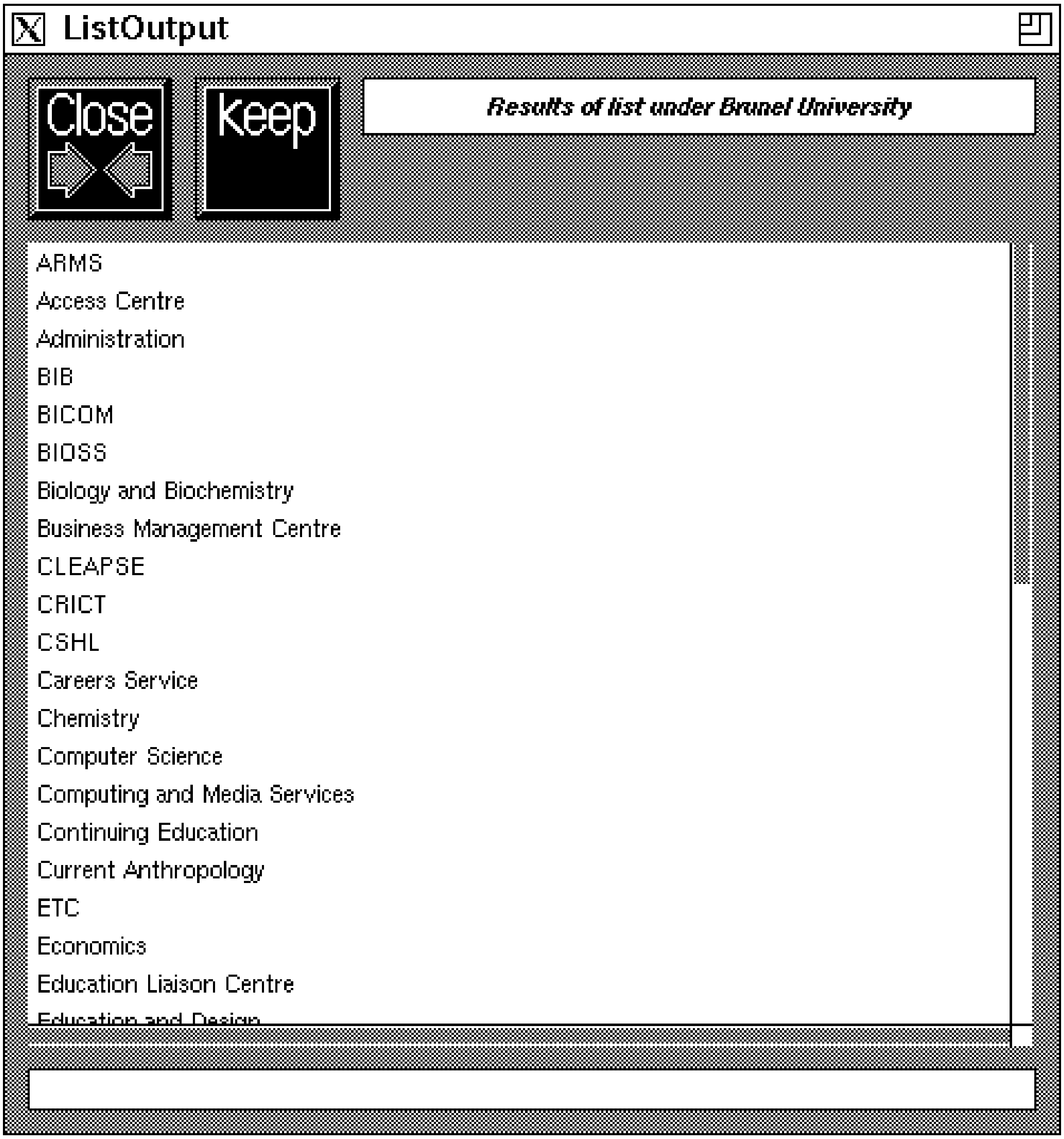
Pod Main Window

When Pod is started, the main window appears. This shows the current directory position, allows a search value to be entered, displays and sets, via a drop down menu, the current search type (in this case "Person"), and allows functions of the the interface to be invoked by pressing buttons.



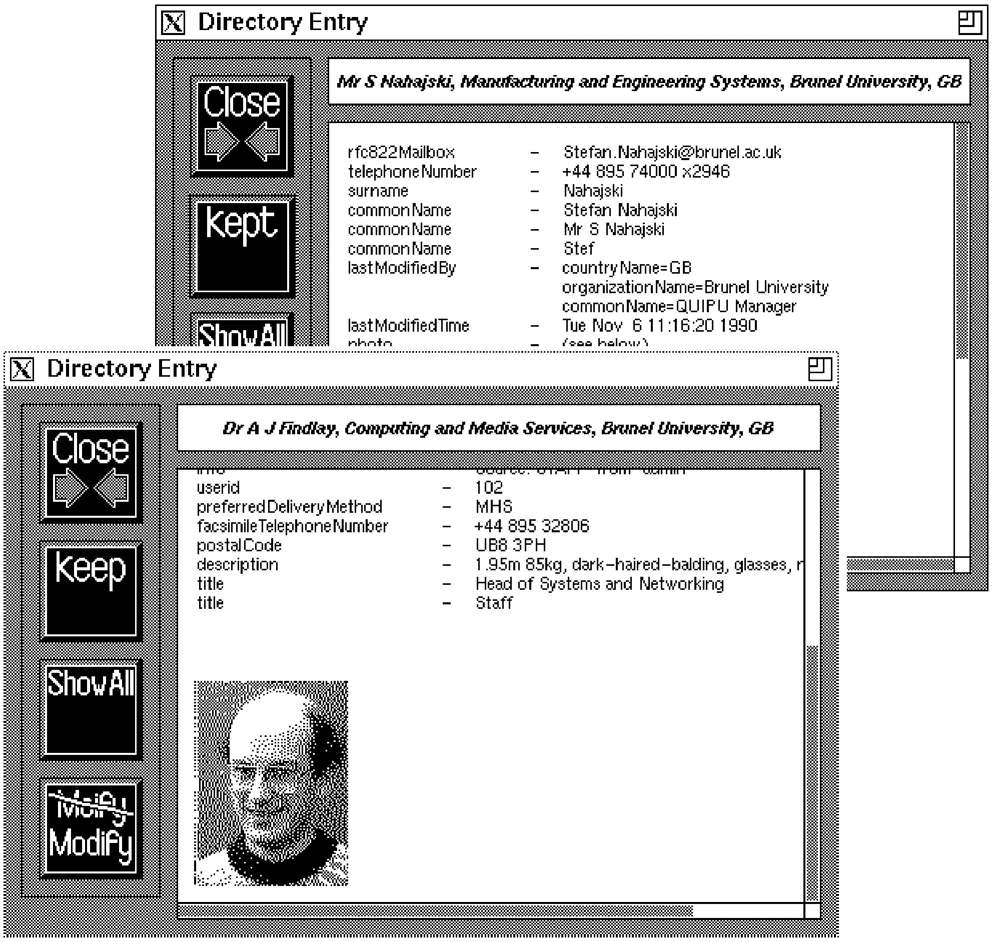
Pod List Pop-up

The list pop-up can contain the result of a list, search or the history list. The contents are described in the status bar (top right), and errors or limits are reported in the bottom status bar.



Pod Read Pop-up

The Keep/Kept button that can be seen in these examples is analogous to the push pin used in Open Look. If a Keep button is pressed, the associated window will not be used for subsequent operations and the Keep label changes to Kept. This makes it possible to collect a number of read (and list) pop-ups on screen at the same time.



5. Design Document

A design document[3] has been produced which reflects the experiences and feedback obtained during the development of the early interfaces. These experiences and the feedback can broadly be classified into two groups; those that reflect a personal preference or suitability for a given context, and those that describe an inconsistency or weakness in the design (although these can also be influenced by the former). The design incorporates both aspects of the feedback received.

The design is divided into two parts describing a visual interface, and a query engine.

5. 1 Visual Interface

The design of user interfaces is fraught with dangers. It is all too easy to annoy or frustrate a user by grouping functionality 'badly', laying out 'wrongly' or over/under specifying the interface for example. The design of the visual interface attempts to avoid many of these dangers by putting such decisions in the hands of the user. This is accomplished by providing a kit of parts suitable for building DUA interfaces.

5.1. 1 Kit of Parts

The kit of parts comprises such visual objects as buttons, lists, directory read areas and also actions such as read, list and search. The parts can be grouped together, layouts can be specified and actions can be associated with objects. A specification language has been defined allowing the configuration of such a interface to be read from a file. At a later stage the configuration file may be machine generated as a result of an interactive configuration session with a user.

5. 2 Query Engine

The query engine provides effective access to and control over data in the Directory, regardless of the visual interface with which it cooperates. In addition to normal directory operations such as read, modify and list, the query engine enables the user to define directory object types. These object types are used as the basis for the search operation. An object type definition specifies the collection of X.500 attributes which denotes the object type, the attributes to match against and the type of matching to perform for each attribute, and an amount of structural information which describes how a given type of entry relates to the rest of the Directory Information Tree (DIT). As an example, the generic type 'Place' might be informally defined as:

```
Type denoted by:
objectClass contains 'locality'
OR objectClass contains 'room'
OR objectClass contains 'country'

Match on:
=countryName
OR ~stateOrProvinceName      # approx
OR %locality                 # substring
OR =friendlyCountryName      # equals

Label:
'Place'

Structural Information:
for entries of type: country locality
   children may be: organization
                    locality
for entries of type: room
   children may be: none
```

The structural information in an object definition provides hints to the type of search strategy that should be employed. Other search strategies, such as User Friendly Naming[4] may also be used where appropriate.

6. Data Manipulation

As has already been mentioned, a high degree of user configurability is an important feature in the design of the directory interface. The configuration of the visual interface and the query engine have been briefly described. However no mention has yet been made of how the data extracted from the Directory will be used. The X500 standard suggests a number of different modes of use. Browsing or simple lookup will form a large part of any use, but there will also be a need to store data for local use in a number of different formats. Typical required formats may be address labels created from a distribution list stored in the Directory, or maybe business cards containing photographs.

The design introduces an idea called 'save formats'. These are user configurable and specify the layout and content of a named format. As an example, consider a save format called *nametag*. This might be defined to contain the photograph, name and address of the given directory entry and laid out say with the text to the right of the photograph. The user could then, whilst using the interface, select the *nametag* save format and then choose people from the Directory for whom name tags should be produced. The exact mechanism by which layout and other post processing will be achieved for the multitude of possible formats has not yet been decided. It is likely however that a save format will specify the content and a post processor which will be invoked to deal with layout and output to a file or other device.

7. Implementation

Implementations of directory interfaces are required to run under DOS and UNIX. Versions suitable for X (using the Athena widget set), X using OSF Motif, and a character addressable terminal are required for UNIX. DOS versions suitable for Microsoft Windows and character mode use are required. The DOS versions are intended for two different scenarios; the Windows implementation is expected to be running on a highly specified DOS machine which will also be running a DOS implementation of the OSI stack, whereas the character mode interface is intended for minimal DOS machines connected to a query engine on a serving DOS machine via a PC LAN.

The success or otherwise of the design described above will depend to some degree on the support and flexibility of the proposed platforms. As mentioned, implementations are required for DOS and UNIX, and the difference in support provided by these two systems has caused major differences in the implementations of the interfaces to appear. For the most part these differences will affect the visual interface part of the design. More particularly, it is expected that the DOS interface will have very restricted configurability. The character mode visual interfaces are not expected to be highly configurable.

8. Future

The work mentioned so far has been aimed at providing access to information the Directory holds regarding people or places. The X.500 standard recognises many different modes of use for the Directory. One mode which is currently of interest is its use as a distributed diary store. Any person or room for example could have a diary attribute associated with them.

This could be interrogated or altered, subject to access controls, and thus meetings or bookings arranged.

Other modes of Directory use will doubtless appear and each will have specific requirements of a user agent.

9. Contacts

The authors are open to any comments or questions which you may have and can be contacted at X500@brunel.ac.uk. The source code for sd, xd and pod are available from Brunel on request.

10. References

[1] ISO/CCITT Recommendation X.500: The Directory - Overview of Concepts, Models and Services, Geneva, March 1988. ISO 9594 is technically aligned with X.500.

[2] The ISO Development Environment User's Manual, Volume 5: QUIPU, UCL, Jan 1990, Kille, Robbins, Roe, Turland.

3 Design Document: Xdir - X.500 Directory User Agent, Andrew Findlay, Damanjit Mahl, Stefan Nahajski, Brunel University, June 1990.

4 Using the OSI Directory to achieve User Friendly Naming, S.E. Kille, June 1990